IN THE CLAIMS

Claims 1-17 are cancelled herein. Claims 18, 26, and 33 have been amended. All pending claims are reproduced below.

1 1. (Cancelled)

1 2. (Cancelled)

1 3. (Cancelled)

1 4. (Cancelled)

1 5. (Cancelled)

1 6. (Cancelled)

1 7. (Cancelled)

1 8. (Cancelled)

1 9. (Cancelled)

1 10. (Cancelled)

1 11. (Cancelled)

1 12. (Cancelled)

1 13. (Cancelled)

14. (Cancelled)

15. (Cancelled)

16. (Cancelled)

17. (Cancelled)

18. (Currently Amended) A method for compiling a functional description expressed in an interpretive, algorithmic language into target code for selected hardware, the method comprising the steps of:

~~receiving~~ parsing the functional description expressed in the interpretive, algorithmic language with at least one undeclared variable into an abstract syntax tree;

inferring a type and dimension for the undeclared variable by analyzing the usage of the undeclared variable in the abstract syntax tree;

assigning ~~a~~ the inferred type and ~~a~~ dimension to the ~~at least one~~ undeclared variable ~~by analyzing the functional description to form an abstract syntax tree~~;

transforming compound statements in the abstract syntax tree into a series of single statements; and

translating the abstract syntax tree into a register transfer level format.

19. (Previously Presented) The method for compiling a functional description of claim 18, further comprising the steps of:

receiving a user directive file including at least one user defined directive selected from the group consisting of constraint directives, assertions, and compiler hints; and

annotating the functional description according to the user directive file.

20. (Previously Presented) The method for compiling a functional description of claim 18, further comprising the steps of:

analyzing a value range of the at least one undeclared variable; and

assigning a required precision for the at least one undeclared variable.

21. (Previously Presented) The method for compiling a functional description of claim 20, further comprising the step of:

parsing a real undeclared variable into an integer part and a fractional part, wherein said real undeclared variable is one of said at least one undeclared variable.

22. (Previously Presented) The method for compiling a functional description of claim 18, further comprising the steps of :

analyzing array access patterns across loop iterations; and

replacing a statement in a loop including a memory access with multiple statements including the memory access to reduce the number of individual memory accesses.

23. (Previously Presented) The method for compiling a functional description of claim 18, further comprising the steps of:

analyzing compound loop structures to identify pipeline opportunities; and

applying the pipeline algorithm to pipeline opportunities to generate nodes corresponding to the loop body, predicate nodes corresponding to loop conditional statements, and a schedule for scheduling pipeline operations.

24. (Previously Presented) The method for compiling a functional description of claim 18, wherein the step of transforming compound statements in the abstract syntax tree into a series of single statements comprises the step of:

expanding a matrix operation into at least one loop.

25. (Previously Presented) The method for compiling a functional description of claim 18, wherein the step of transforming compound statements in the abstract syntax tree into a series of single statements comprises the step of:

deconstructing a compound statement into at least one simple statement.

26. (Currently Amended) A system for compiling a functional description expressed in an interpretive, algorithmic language into target code for selected hardware comprising:

a parser for ~~receiving~~ parsing the functional description expressed in the interpretive, algorithmic language with at least one undeclared variable into an abstract syntax tree;

a type-shape analyzer, coupled to the parser, for ~~assigning~~ inferring a type and a dimension to the ~~at least one~~ undeclared variable by analyzing use of the undeclared variable in the abstract syntax tree ~~the functional description to form an abstract syntax tree~~;

a statement deconstructor, coupled to the type-shape analyzer, for transforming a compound statement in the abstract syntax tree into at least one simple statement; and

a translator, coupled to the statement deconstructor, for translating the abstract syntax tree into a register transfer level format.

27. (Previously Presented) The system for compiling a functional description of claim 26, further comprising:

a user directive file, coupled to the parser, for annotating the functional description with at least one user defined directive selected from the group consisting of constraint directives, assertions, and compiler hints.

28. (Previously Presented) The system for compiling a functional description of claim 26, further comprising:

a precision analyzer, coupled to the type-shape analyzer, for determining the precision of the at least one undeclared variable.

29. (Previously Presented) The system for compiling a functional description of claim 28, further comprising:

a real number parser, coupled to the precision analyzer, for parsing a real number into an integer part and a fractional part.

30. (Previously Presented) The system for compiling a functional description of claim 26, further comprising:

a memory access optimizer, coupled to the statement deconstructor, for analyzing array access patterns across loop iterations and replacing a statement in a loop including a memory access with multiple statements including the memory access to reduce the number of individual memory accesses.

31. (Previously Presented) The system for compiling a functional description of claim 26, further comprising:

a pipeline optimizer, coupled to the statement deconstructor, for analyzing compound loop structures to identify pipeline opportunities and applying the pipeline algorithm to pipeline opportunities to generate nodes corresponding to the loop body, predicate nodes corresponding to loop conditional statements, and a schedule for scheduling pipeline operations.

32. (Previously Presented) The system for compiling a functional description of claim 26, wherein the statement deconstructor for transforming a compound statement in the abstract syntax tree into at least one simple statement comprises:

a scalarizer, coupled to the type-shape analyzer, for expanding a matrix operation into at least one loop.

33. (Currently Amended) One or more computer readable storage devices having computer readable code embodied on said computer readable storage device, said computer readable code for programming one or more computers to perform a method for compiling a functional description expressed in an interpretive, algorithmic language into target code for selected hardware, the method comprising the steps of:

~~receiving~~ parsing the functional description expressed in the interpretive, algorithmic language with at least one undeclared variable into an abstract syntax tree;

inferring a type and dimension for the undeclared variable by analyzing the usage of the undeclared variable in the abstract syntax tree;

assigning ~~a~~ the inferred type and ~~a~~ dimension to the ~~at least one~~ undeclared variable ~~by analyzing the functional description to form an abstract syntax tree~~;

transforming compound statements in the abstract syntax tree into a series of single statements; and

translating the abstract syntax tree into a register transfer level format.

34. (Previously Presented) One or more computer readable storage devices having computer readable code embodied on said computer readable storage device, said computer readable code for programming one or more computers to perform a method for compiling a functional description of claim 33, further comprising the step of:

receiving a user directive file including at least one user defined directive selected from the group consisting of constraint directives, assertions, and compiler hints; and

annotating the functional description according to the user directive file.

35. (Previously Presented) One or more computer readable storage devices having computer readable code embodied on said computer readable storage device, said computer readable code for programming one or more computers to perform a method for compiling a functional description of claim 33, further comprising the step of:

analyzing a value range of the at least one undeclared variable; and

assigning a required precision for the at least one undeclared variable.

36. (Previously Presented) One or more computer readable storage devices having computer readable code embodied on said computer readable storage device, said computer readable code for programming one or more computers to perform a method for compiling a functional description of claim 33, further comprising the step of:

analyzing array access patterns across loop iterations; and

replacing a statement in a loop with a memory access with multiple statements with the memory access to reduce the number of individual memory accesses.

37. (Previously Presented) One or more computer readable storage devices having computer readable code embodied on said computer readable storage device, said computer readable code for programming one or more computers to perform a method for compiling a functional description of claim 33, further comprising the step of:

analyzing compound loop structures to identify pipeline opportunities; and

applying the pipeline algorithm to pipeline opportunities to generate nodes corresponding to the loop body, predicate nodes corresponding to loop conditional statements, and a schedule for scheduling pipeline operations.